Drift Detection in Text Data with Document Embeddings

Robert Feldhans[1], Adrian Wilke[2], Stefan Heindorf[2], Mohammad Hossein Shaker[3], Barbara Hammer[1], Axel-Cyrille Ngonga Ngomo[2], and Eyke Hüllermeier[3]

[1] Bielefeld University, Germany
{rfeldhans,bhammer}@techfak.uni-bielefeld.de
[2] DICE group, Department of Computer Science, Paderborn University, Germany
{adrian.wilke,heindorf,axel.ngonga}@uni-paderborn.de
[3] University of Munich (LMU), Germany
mhshaker@mail.uni-paderborn.de,eyke@ifi.lmu.de

Abstract. Collections of text documents such as product reviews and microblogs often evolve over time. In practice, however, classifiers trained on them are updated infrequently, leading to performance degradation over time. While approaches for automatic drift detection have been proposed, they were often designed for low-dimensional sensor data, and it is unclear how well they perform for state-of-the-art text classifiers based on high-dimensional document embeddings. In this paper, we empirically compare drift detectors on document embeddings on two benchmarking datasets with varying amounts of drift. Our results show that multivariate drift detectors based on the Kernel Two-Sample Test and Least-Squares Density Difference outperform univariate drift detectors based on the Kolmogorov–Smirnov Test. Moreover, our experiments show that current drift detectors perform better on smaller embedding dimensions.

Keywords: Drift Detection · Document Embeddings · BERT · Word2Vec

1 Introduction

One of the key challenges when deploying machine learning models in practice is their degradation of performance after having been deployed [2]. In addition to technical issues, e.g., changes in the data format, performance degradation can be caused by (1) drift in the class distribution (virtual drift), and (2) drift in the labels (real drift). In the former case, the model might have insufficient training data for all classes and assumes a wrong prior probability. In the latter case, similar data points are labeled differently over time. For example, in the domain of natural language processing, this boils down to using language differently over time (virtual drift) due to novel words, grammatical constructs and writing styles. It also affects changing class labels over time (real drift) due to new human annotators joining the teams or updated annotation guidelines, e.g., changing the definition of disinformation [15], harassment [27], sexism [8], clickbait [7], etc.

To detect drift automatically and notify machine learning engineers to potentially update their models, automatic drift detectors have been proposed [12, 17, 22]. However, as we show in this paper, state-of-the-art drift detectors are hardly applicable to modern NLP applications: they assume the input data to be low-dimensional, hand-engineered vector spaces, while modern NLP applications employ complex language models such as transformers with high-dimensional, latent document embeddings. Towards this end, we empirically compare state-of-the-art drift detectors which have mainly been designed for low-dimensional vector spaces and we apply them to drift detection in natural language texts:

RQ1: Which drift detector works best for document embeddings?
RQ2: How does the performance of a drift detector depend on the embedding dimensions?

We perform our experiments using two benchmarking datasets into which we inject different amounts of drift. Moreover, we investigate how the predictive performance of a drift detector depends on the embedding dimensions. Our experiments show that multivariate approaches based on the Kernel Two-Sample test (KTS) and Least-Squares Density Difference (LSDD) outperform univariate drift detectors based on the Kolmogorov-Smirnov test (KS) and that most approaches perform better on low-dimensional data. The code underlying our research is publicly available.[4]

The remainder of this article is structured as follows: In Section 2, we briefly outline related work. Section 3 describes our methodology, i.e., datasets and drift detectors as well as our evaluation setup. Finally, Section 4 presents our results and Section 5 our final conclusions.

2 Related Work

Concept drift has become a highly researched field. There are several recommendable introductions from Lu et al. [18], Gama et al. [11], Nishida and Yamauchi [21], Gama and Castillo [10] and Basseville and Nikiforov [4]. Gama et al. [11] conducted a survey that provides an introduction to concept drift adaptation, which includes patterns of changes over time (e.g., incremental changes) and evaluation metric criteria (e.g., probability of true change detection and delay of detection), which are part of our injection experiments. Baier et al. [3] provided an analysis of 34 articles related to concept drift and a framework of 11 categories to characterize predictive services like drift detectors. With respect to their overview of existing approaches, we fill the gap in the category *data input* (here: unstructured text); our paper can be classified as being *gradual* as well as *sudden* in the category *type of change*. The overview of concept drift by Tsymbal [24] distinguished between sudden and gradual concept drift. Additionally, batch systems (here: LSDD, MMD) and online systems (here: Confidence Distribution Batch Detection, CDBD) are distinguished. The classical FLORA systems [25] as well as the more recent MDEF [13], ADWIN [5], and EDDM Baena-Garcıa

[4] https://github.com/EML4U/Drift-detector-comparison

et al. [1] are also worth mentioning. Moreover, drift detection has successfully been used on image data [17, 22] and various other applications [28].

3 Comparison of Drift Detectors

Given a classification problem, let X be a feature vector, y be the target variable and $P(y, X)$ their joint distribution. Following Gama et al. [11], *real concept* drift refers to changes in $P(y|X)$ and *virtual drift* refers to changes in $P(X)$, i.e., if the distribution of the incoming data changes. In our case, we focus on virtual drift. One of the particular challenges for natural language processing is that texts are typically transformed to high-dimensional feature vectors X which might only appear once in the dataset. Further, state-of-the-art NLP models, such as neural networks and transformers, follow a discriminative paradigm [20] instead of a generative one, making it difficult to estimate probability distributions.

For the experiments on drift detectors (Sec. 3.1), we use two real-world datasets that are transformed using three embedding models—one BERT model [9] with 768 dimensions, one Word2Vec model [19] with 768 dimensions, and another Word2Vec model with 50 dimensions—to explore the effect of different numbers of dimensions (Sec. 3.2). Finally, the data are arranged in four subsets in preparation for the following experiments (Sec. 3.3).

3.1 Drift Detectors in the Experiments

We selected the popular drift detectors KS, KTS and additionally, a more recent approach, LSDD (2018), as well as the semi-supervised CDBD.

Kolmogorov-Smirnov (KS) is a statistical test for agreement between two probability distributions using the maximum absolute difference between the distributions. We use the feature-wise two-sample implementation of the Alibi Detect[5] library. For multivariate data, Bonferroni correction is used to aggregate the p-values per dimension.

Kernel Two-Sample (KTS) [12] is a statistical independence test based on Maximum Mean Discrepancy (MMD). MMD is the squared distance between the embeddings of two distributions p and q in a *reproducing kernel Hilbert space*, $MMD(p, q) = ||\mu_p - \mu_q||^2_{\mathcal{H}}$, where μ denotes the mean embeddings. We use the implementation of Emanuele Olivetti[6] for our experiments.

Least-Squares Density Difference (LSDD) [6] is based on the least-squares density difference estimation method. For two distributions p and q, it is defined as $LSDD(p, q) = \int (p(x) - q(x))^2 \, dx$. To apply the test, we utilize the Alibi Detect[5] implementation.

Confidence Distribution Batch Detection (CDBD) [16] is an uncertainty-based drift detector using a two-window paradigm coupled with Kullback-Leibler divergence applied to a confidence score. It can be used with any

[5] https://github.com/SeldonIO/alibi-detect
[6] https://github.com/emanuele/kernel_two_sample_test

classifier that produces a confidence (uncertainty) score about the classifier predictions. Given that CDBD requires labels in the beginning for training the classifier, it is the only semi-supervised detector in our comparison. CDBD compares the divergence between the distribution of confidence scores of a batch of reference instances to test instances. The higher the divergence, the more drift there is between the reference batch and the test batch. In this paper, we used Random Forest as the classifier and entropy of the output probability distributions as the confidence score.

3.2 Source Datasets and Embedding Models

Amazon Movie Reviews The Amazon Movie Review dataset[7] consists of nearly 8 million user reviews, from 1997 to 2012, of movies purchasable on the Amazon website. In particular, this dataset contains the joined user reviews and summaries thereof in text form, which we will use to detect drift, and a score from one to five that the user gave the movie. The average length of each text is 172 words. It can be noted that this score should be directly correlated to the texts the user wrote and be indicative of the sentiment of said texts. As we suspect some (uncontrolled) drift over the twelve-year time frame of the dataset, we opted to use the data of one year (i.e., 2011) to reduce possible changes over time. We generally use the scores of the reviews as classes, especially for retraining the models.

The BERT model used with this dataset is a retrained version of the pretrained model 'bert-base-uncased' provided by the *Hugging Face* [26] library. The pretrained BERT model was retrained for nine epochs, using all entries from 2011, and provides embeddings with 768 dimensions.

Both BoW models were computed using all tokenized texts contained in the dataset. For the training, we used 40 epochs and a minimum count of 2 for each word to be included. As the training algorithm, a distributed bag of words (PV-DBOW) of the *Gensim*[8] 3.8.3 software was applied. The computation took 32 hours (50 dim) and 44 hours (768 dim) for Amazon and 1.5 hours for the two Twitter models on a 4x Intel(R) Xeon(R) CPU E5-2695 v3 @ 2.30GHz machine with no GPU and 64 GB RAM.

Twitter Election The Twitter Election dataset[9] is composed of tweets that refer to the two candidates of the 2020 US presidential election (i.e., these tweets use the hashtag #Biden and/or #Trump). All given data points were created during the last three weeks before the election, i.e., t_{min} = 2020-10-14 to t_{max} = 2020-11-07. The original dataset contains around 1.7 million singular data points, of which ca. 777k use hashtag #Biden and ca. 971k use #Trump.

There are two distinct points in time which coincide with real-world events and are thus suspected to differ in distribution. These events are the last TV debate

[7] https://snap.stanford.edu/data/web-Movies.html
[8] https://radimrehurek.com/gensim/
[9] https://www.kaggle.com/manchunhui/us-election-2020-tweets

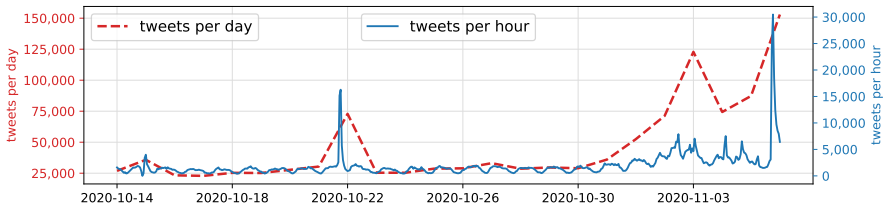

Fig. 1. Amount of tweets in the Twitter election dataset by day (red) and hour (blue). Note the sharp spike during the last TV debate (t_{debate} = 2020-10-22) and the overall increase during the election day ($t_{election}$ = 2020-11-03). Slight inaccuracies in the timeline can be attributed to time zone-related shifts.

before the election (t_{debate} = 2020-10-22) and the election day itself ($t_{election}$ = 2020-11-03). The increased number of data points in both of these time frames supports this claim, as can be seen in Fig. 1. We generally distinguish #Biden and #Trump as the two classes of this dataset. As such, we removed all ambiguous tweets that contain both the #Biden and #Trump hashtags. Some data points in this set contain non-English language, most notably Spanish. To reduce the effect of non-English data points, we removed tweets from the dataset which were detected as non-English with the Python implementation[10] of *langdetect*.[11] Around 500k data points were removed this way, resulting in a dataset of ca. 521k #Biden tweets and 680k #Trump tweets.

The BERT model used with this dataset is a pretrained model provided by the *Hugging Face* library called 'bert-base-multilingual-cased'. It was chosen because of its ability to handle multilanguage input data, as the dataset contains traces of multi-language data. This model was not retrained.

The BoW models were computed analogously to the ones used in the Amazon movie dataset and took 1.5 hours to compute on the same machine as used for the Amazon models.

3.3 Evaluation Setup: Sampling

To evaluate the drift detector's sensitivity and specificity, we created several subsets of the datasets with precisely controlled drift.

Drift Induction This subset consists of two balanced sets of 2,000 samples of randomly chosen but class-balanced data points. The first set is then gradually injected with specifically chosen negative adjectives. In each step, a certain percentage $\gamma_i \in \{0.05 \cdot i | i = 0, 1, 2, ..., 20\}$ of texts is injected with one of these negative adjectives. With this subset, we want to evaluate the speed and confidence with which drift detectors detect gradually induced drift. Injection is done between two randomly chosen words of the text, and each text is injected with a maximum

[10] https://pypi.org/project/langdetect/
[11] http://code.google.com/p/language-detection/

of one word over all steps. For example, in step γ_2, 0.1=10% of texts are injected with at most one negative adjective. This resembles the method of Shoemark et al. [23], with the difference that we are not replacing but adding real words instead of made-up ones.

The specific list of these adjectives was obtained by choosing the 22 adjectives that occur at least 500 times in all Amazon movie reviews of both one or five stars and appear at least twice as often in one star reviews. To ensure these adjectives are negatively connotated, we used a list of 4,783 negative opinion words,[12] mined from negative customer reviews [14]. The experiments using this subset are repeated ten times using unique data for each run.

Twitter Election Specifics In processing this dataset, we also use the negative adjectives based on the Amazon Movie Review dataset for injection, to better compare the Amazon and Twitter variants. Additionally, there is a lack of a distinct gradient (akin to the score) between both classes of the dataset, so an analogous but distinct approach to the Amazon variant was discarded.

Same Distribution Using this subset, we test the drift detectors against data drawn from the same distribution to explore whether the drift detectors abstain from identifying drift where there is none. This is of importance in general applications of drift handling, as retraining a model to compensate for drift is often expensive, so it should only be done if necessary. This subset consists of 500 random samples per class (i.e., 2,500 samples for the Amazon and 1,000 samples for the Twitter dataset). It is tested against twenty more subsets created with the same criteria, but distinct samples. We then present the mean of the results.

Different Classes This subset is used to evaluate the drift between data of two different classes. As such, its main purpose is to establish a tangible maximum of (virtual) drift possible in each dataset, test whether or not the drift detectors are able to detect it, and to give context for the drift induction subset. With this in mind, any drift detector should be able to detect the virtual drift in this experiment. For this, 1,000 samples were taken from each class. The test was repeated ten times with unique data.

Amazon Movie Reviews Specifics The classes used for this dataset were those defined by reviews of scores one and five.

Different Distribution In this subset, we apply the prior knowledge of the Twitter dataset, thus no Amazon variant of this subset exists. In contrast to the other subsets, where drift is set up artificially, this one presents an application of a real-world example of drift. We check for drift between data of a typical point in time ($t_{reference} = t_{min} + 100h$) and three other points of interest, i.e., another typical point $t_{base} = t_{reference} + 24h$ to establish a baseline, t_{debate} and $t_{election}$.

[12] https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon

For each t, 1,000 class-balanced samples were taken between t and $t + 24h$. For this subset, we generate 8 permutations of this setup and present the mean of the results.

Semi-supervised Algorithm The semi-supervised algorithm CDBD requires a deviant structuring of input data, as a portion of the data given to the detector is used to train the model. To avoid a small reference batch, the permutations of CDBD are different in that the entire data given to the detector is randomly shuffled to train different models and create different reference batches for each permutation.

4 Results

Results of the Drift Injection Experiments

Based on the drift induction setup (Sec. 3.3), we evaluate the speed and confidence with which drift detectors detect slowly induced drift in this experiment. An ideal drift detector would produce a high p-value with little to no injected words but quickly drop when more words are injected. The resulting p-values of the experiments are displayed in Fig. 2.

Regarding **RQ1**, the KTS and LSDD detectors generally produce similar results while outperforming CDBD and KS. KTS produced better results than LSDD, especially for the high-dimensional BoW experiment. In particular, KS behaves conservatively in its estimations and struggles to detect drift, even with considerable injection of words. CDBD shows erratic behavior with all BoW embeddings; it outperforms KS only with the BoW-768 embeddings and on the Twitter dataset with BoW-50 embeddings.

Regarding **RQ2**, the lower-dimensional BoW-50 data generally performs better than or equal to both higher-dimensional sets, and the best detectors (i.e., KTS and LSDD) reach a p-value of 0.05 with less induced drift than with the higher-dimensional data. However, it is important to note that almost all detectors start with a much higher p-value on the higher-dimensional BERT data, as described in Sec. 3.3.

Results of Same Distribution Experiments

In this experiment, we evaluate the drift detectors' ability to not falsely detect drift where none is present. A higher p-value suggests a better performance (see Tab. 1).

With respect to **RQ1**, the CDBD detector generally outperforms the other detectors across all embeddings ($p \approx 0.7$). The KS detector performs very consistently ($p \approx 0.5$). Depending on the model and embedding method used, the KTS yields the single best results in this test (on Amazon BERT) but has a high standard deviation. LSDD's performance is generally about as good as KTS's on each dataset, although slightly better on average. Its standard deviation is

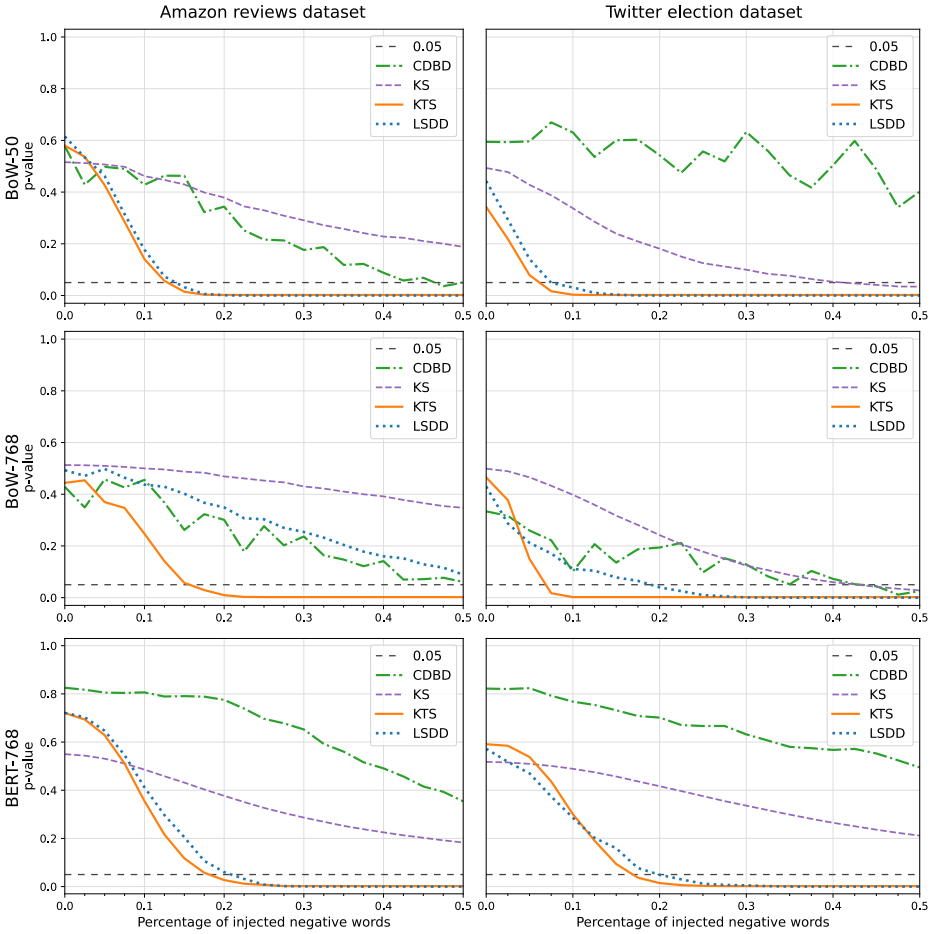

Fig. 2. Drift detection results on injection experiments: Amazon reviews dataset (left) and Twitter election dataset (right) as well as the models BoW-50 (top), BoW-768 (middle) and BERT-768 (bottom). Presented is the mean p-value of all runs.

comparable to KTS's. Regarding **RQ2**, we do not see a consistent effect of the embedding dimension.

Results of the Different Class Experiments

In this experiment, we evaluate the drift detectors based on their ability to detect the maximum possible (virtual) drift in each dataset. Here, a lower p-value suggests a better performance (see Tab. 2). The CDBD detector is omitted from this experiment since it is impossible to train a classifier on single-classed data given only the data used in this experiment.

With regard to **RQ1**, both LSDD and KTS perform as expected and consistently provide nearly perfect results across both datasets and all embeddings.

Table 1. Drift detector scores of same distribution exp. (mean p-value of all runs, higher ∅ is better)

Dataset	Model	CDBD		KS		KTS		LSDD	
		∅	stdev	∅	stdev	∅	stdev	∅	stdev
Amazon	BoW-50	**0.6494**	0.2304	0.4678	0.0397	0.4294	0.2920	0.5060	0.2990
Amazon	BoW-768	**0.6338**	0.0727	0.5009	0.0224	0.5860	0.3010	0.4140	0.3492
Amazon	BERT-768	0.7148	0.1544	0.5676	0.0195	**0.8666**	0.1372	0.8160	0.2051
Twitter	BoW-50	**0.7863**	0.0785	0.5330	0.0419	0.5930	0.2398	0.5980	0.2392
Twitter	BoW-768	0.4257	0.1190	0.5204	0.0208	0.4828	0.3100	**0.6380**	0.2825
Twitter	BERT-768	**0.7720**	0.1448	0.5114	0.0190	0.3878	0.1940	0.4910	0.2048

The KS detector struggles to detect drift with statistical significance at the 0.05 significance level. It is able to do so only on the Amazon BERT embeddings.

Regarding **RQ2**, the results show a tendency of drift detectors to perform better for lower embedding dimensions, as exemplified by the KS detector.

Table 2. Results of the different class experiments (mean p-value of all runs, lower ∅ is better)

Dataset	Model	KS		KTS		LSDD	
		∅	stdev	∅	stdev	∅	stdev
Amazon	BoW-50	0.0709	0.0145	0.0020	0.0000	**0.0000**	0.0000
Amazon	BoW-768	0.0870	0.0160	0.0020	0.0000	**0.0000**	0.0000
Amazon	BERT-768	0.0126	0.0012	0.0020	0.0000	**0.0000**	0.0000
Twitter	BoW-50	0.1009	0.0207	0.0020	0.0000	**0.0000**	0.0000
Twitter	BoW-768	0.2522	0.0266	**0.0020**	0.0000	0.0180	0.0218
Twitter	BERT-768	0.1205	0.0047	0.0020	0.0000	**0.0000**	0.0000

Results of the Twitter Different Distribution Experiments

In this experiment, we evaluate the drift detectors in a controlled scenario with prior information about the dataset. A higher p-value in t_{base} and lower p-values in t_{debate} and $t_{election}$ suggest a better performance (see Fig. 3 and Tab. 3). For this experiment, we do not report CDBD results since large fluctuations in p-values render CDBD unusable in our experimental setup, emphasizing the necessity of large datasets for supervised drift detectors.

Regarding **RQ1**, nine drift detection results are available per detector: three kinds of embeddings, each with three points in time. LSDD correctly predicted drift in eight of the nine cases, KTS in seven cases and KS in six cases. From a qualitative perspective, the KS detector produces the most pronounced curve (see Fig. 3), i.e., the largest difference in p-values with respect to t_{base}, and correctly

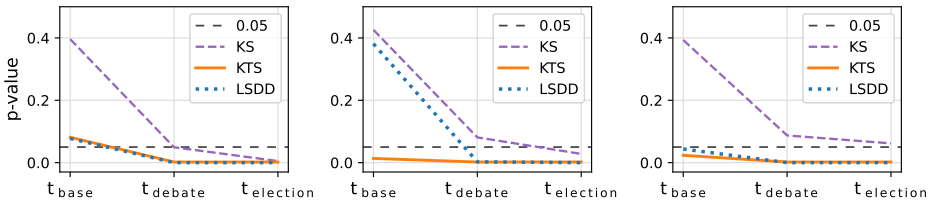

Fig. 3. Results of Twitter different distribution experiment (mean p-value of all runs): BoW-50 (left), BoW-768 (center) and BERT-768 (right)

predicts no drift in t_{base} across all embedding models. However, it struggles to detect drift in t_{debate} and $t_{election}$ at the 0.05 significance level. Both the KTS and LSDD detectors are capable of detecting drift that occurred in those points of time.However, their mean p-values are close to the 0.05 significance level in t_{base}, leading to fluctuating decisions considering their standard deviation.

For **RQ2**, all detectors produced correct results for the low-dimensional BoW-50 embeddings, whereas this is not guaranteed for higher dimensions.

Table 3. Results of the different distribution experiments (mean p-value of all runs)

t	Model	KS		KTS		LSDD	
		∅	stdev	∅	stdev	∅	stdev
t_{base}	BoW-50	**0.3961**	0.0697	0.0813	0.1444	0.0775	0.0935
t_{base}	BoW-768	**0.4259**	0.0225	0.0135	0.0304	0.3812	0.2382
t_{base}	BERT-768	**0.3933**	0.0274	0.0235	0.0187	0.0437	0.0394
t_{debate}	BoW-50	0.0493	0.0446	0.0020	0.0000	**0.0000**	0.0000
t_{debate}	BoW-768	0.0810	0.0853	**0.0020**	0.0000	0.0025	0.0066
t_{debate}	BERT-768	0.0873	0.0674	0.0020	0.0000	**0.0000**	0.0000
$t_{election}$	BoW-50	0.0052	0.0029	0.0020	0.0000	**0.0000**	0.0000
$t_{election}$	BoW-768	0.0281	0.0137	0.0020	0.0000	**0.0000**	0.0000
$t_{election}$	BERT-768	0.0618	0.0104	0.0020	0.0000	**0.0000**	0.0000

5 Conclusion

Regarding our research questions, our conclusions are as follows:

RQ1. Our experimental results suggest LSDD and KTS as the best drift detectors with LSDD slightly outperforming KTS in the real-world Twitter election experiment. KS produced rather average results in all experiments due to its conservative estimation of p-values. CDBD, as a supervised drift detector, requires a large reference batch to produce robust results, questioning its usefulness in many practical applications.

RQ2. Our results indicate that lower embedding dimensions tend to produce better drift detection results.

In future work, we would like to further explore the effect of different dimensionality reduction techniques on drift detectors and to devise novel drift detectors specifically tailored to text data with high-dimension document embeddings, e.g., based on the similarity metrics employed by the embedding approaches.

Acknowledgments This work has been supported by the German Federal Ministry of Education and Research (BMBF) within the project EML4U under the grant no 01IS19080 A and B.

Bibliography

[1] Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno, R.: Early drift detection method. In: Fourth international workshop on knowledge discovery from data streams, vol. 6 (2006)

[2] Baier, L., Jöhren, F., Seebacher, S.: Challenges in the deployment and operation of machine learning in practice. In: ECIS (2019)

[3] Baier, L., Kühl, N., Satzger, G.: How to cope with change? - preserving validity of predictive services over time. In: HICSS, ScholarSpace (2019)

[4] Basseville, M., Nikiforov, I.V.: Detection of Abrupt Changes: Theory and Application. Prentice Hall (1993)

[5] Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: SDM, pp. 443–448, SIAM (2007)

[6] Bu, L., Alippi, C., Zhao, D.: A pdf-free change detection test based on density difference estimation. IEEE Trans. Neural Networks Learn. Syst. **29**(2) (2018)

[7] Chen, Y., Conroy, N.J., Rubin, V.L.: Misleading online content: Recognizing clickbait as "false news". In: WMDD@ICMI, pp. 15–19, ACM (2015)

[8] Chowdhury, A.G., Sawhney, R., Shah, R.R., Mahata, D.: #youtoo? detection of personal recollections of sexual harassment on social media. In: ACL, pp. 2527–2537 (2019)

[9] Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR (2018)

[10] Gama, J., Castillo, G.: Learning with local drift detection. In: ADMA, Lecture Notes in Computer Science, vol. 4093, pp. 42–55, Springer (2006)

[11] Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4), 44:1–44:37 (2014)

[12] Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.J.: A kernel two-sample test. J. Mach. Learn. Res. **13**, 723–773 (2012)

[13] Heit, J., Liu, J., Shah, M.: An architecture for the deployment of statistical models for the big data era. In: IEEE BigData (2016)

[14] Hu, M., Liu, B.: Mining and summarizing customer reviews. In: KDD, pp. 168–177, ACM (2004)

[15] Kumar, S., West, R., Leskovec, J.: Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In: WWW, ACM (2016)
[16] Lindstrom, P., Namee, B.M., Delany, S.J.: Drift detection using uncertainty distribution divergence. Evol. Syst. **4**(1), 13–25 (2013)
[17] Lopez-Paz, D., Oquab, M.: Revisiting classifier two-sample tests. In: ICLR (Poster), OpenReview.net (2017)
[18] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE Trans. Knowl. Data Eng. **31**(12) (2019)
[19] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR (Workshop Poster) (2013)
[20] Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In: NIPS, pp. 841–848, MIT Press (2001)
[21] Nishida, K., Yamauchi, K.: Detecting concept drift using statistical testing. In: Discovery Science, Lecture Notes in Computer Science, vol. 4755, pp. 264–269, Springer (2007)
[22] Rabanser, S., Günnemann, S., Lipton, Z.C.: Failing loudly: An empirical study of methods for detecting dataset shift. In: NeurIPS (2019)
[23] Shoemark, P., Liza, F.F., Nguyen, D., Hale, S.A., McGillivray, B.: Room to glo: A systematic comparison of semantic change detection approaches with word embeddings. In: EMNLP/IJCNLP, pp. 66–76, Association for Computational Linguistics (2019)
[24] Tsymbal, A.: The problem of concept drift: definitions and related work. Computer Science Department, Trinity College Dublin **106**(2), 58 (2004)
[25] Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine learning **23**(1), 69–101 (1996)
[26] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: EMNLP (Demos), pp. 38–45, ACL (2020)
[27] Yin, D., Xue, Z., Hong, L., Davison, B.D., Kontostathis, A., Edwards, L.: Detection of harassment on web 2.0. Proceedings of the Content Analysis in the WEB **2**, 1–7 (2009)
[28] Žliobaitė, I., Pechenizkiy, M., Gama, J.: An overview of concept drift applications. Big data analysis: new algorithms for a new society (2016)